Research on Optimization of memory pool management for high concurrent service requests

LIU Pingping, LU Zhaopan

School of Computer Science and Engineering

Xi'an Technological University, Xi'an 710021,China

Email:1341369601@qq.com

Abstract. In order to quickly and accurately return the information to the user after the keyword are entered, and to effectively reduce the effect on the performance of the program when the search system allocates and deal locates memory frequently under the high concurrency, the Recoverable Fixed Length Memory Pool, Recoverable Variable Length Memory Pool and Allocate Not Free Memory Pool were designed. According to the different scenes features of the search engine. The result shows that, compared with the default system memory allocator, the efficiency of the Recoverable Fixed Length Memory Pool is increased by 70.20% ,the efficiency of the Recoverable Variable Length Memory Pool is increased by 13.84% and the efficiency of the Allocate Not Free Memory Pool is increased by 90.80%.

Keywords: High Concurrency, search engine, memory pool, distributor

1. Introductiong

The search engine is one of the most important applications of the Internet, which involved in information retrieval, distributed processing, semantic web, data mining etc. The reasonable data structure design, the index and the high concurrent system structure are all the factors that influence the query speed. The basic principle of the search engine has been very stable, but in terms of service, quality and performance needs to be optimized.

Most of traditional search engines use keyword matching mode, the system manages memory when the application is not released too frequently, but in the face of massive data processing and storage, search engines seem powerless. There are some drawbacks that directly using the system call Malloc/ Free and New/Delete[1] to distribute and release the memory. For example, calling the Malloc/New system in accordance with the "first match" and "best match" or other algorithms in free memory block table to find a free memory, the memory usage is not high; The system may need to merge free memory blocks when Free / Delete is called, which will result in extra time and space overhead; It is easy to produce a large number of memory fragments when used frequently, which reduces the efficiency and stability of the program; Memory more prone to leaks[2] that caused by memory size continues to increase and memory exhausted. For memory allocation problem, Wang Xiaoyin, a professor of Xi' an University of Posts and telecommunications, analysis and research about the method and principle of the establishment of the memory pool in the article of Implementation and Application of the Memory Pool in Linux Kernel[3]. Memory allocated in memory pool does not need to release, it will be released

when the memory pool is destroyed. Avantages:It speeds up memory allocation, when block of memory is enough, only conduct simple operation such as size judgment and pointer offset; Small memory payloads are high, require less additional information; The memory pool allocated memory usually do not need a separate release, but a unified recovery; In addition to using memory allocation functions instead of malloc, no other special conventions are used.

Therefore, to compare of the traditional search engine memory allocation and the memory poolallocation. In this paper, different memory pools are designed for different application scenarios, it managememory allocation to get the fastest allocate and release speed. For the user’s query, the system’s memory management is completely taken over by the programmer, which is more conducive to investigate problem and optimize system, and quickly return a satisfied result for customer.

2. Pinciples and Key Technologies of Search Engine

Search engine is based on the information extracted from the web site to establish the database, search the relevant records of user query condition matching, and then return the results to the user according to a certain order. The working principle[4] of search engine is divided into four steps: First, using web crawler technology[5] to automatically grab the web page from the Internet, then analysis the original web page, and set up an index database, and finally searching and sorting in the index database .When there are multiple threads operating, if the system only has one CPU, it can not be carried out more than one thread at the same time, it can only divided the running time of CPU into several periods, then allocate the period of time to each thread, a thread code is run in a time, other threads are hanged up, this way we call concurrency. In the condition of high concurrency, the search system frequently allocate and recover memory will degrade the performance of program and the memory is used in a particular way, and pay the cost of performance on the function that is not required.

For long-running background service system, the performance decrease mainly due to the default memory management is a universal, and general memory management usually consider many factors, including the thread, size, recovery time, distribution, frequency and so on. For this reason, it is common to consider use of the memory pool to manage memory allocation, rather than simply using New/Delete, Malloc/Free for dynamic memory allocation. By designing a dedicated memory pool to allocate specific memory and optimize performance in different search application scenarios of search system, and to enhance the mass data storage and search speed, in order to solve the problem of universal memory.

2.1 Principle of memory pool

Memory pool[6] is a way of memory allocation, is a device that can dynamically allocate memory. It can only be used by a specific kernel component (that is, the owner of the pool). Owners usually do not directly use the memory pool, when the common memory allocation fails, the kernel call a particular memory pool function to extract the memory pool, in order to get the extra memory. So the memory pool is only a memory of the kernel memory, used at a specific time.

As shown in figure 1, the memory pool contains a total of 4 memory blocks. When the memory pool is initially generated, only one block of memory is applied to the system, and the returned pointer acts as the head of the entire memory pool. After the application of the continuous demand for memory, memory pool judgment need to dynamically expand, then once again to apply for a new memory block of the system, and all of these memory blocks linked by pointers.

For the operating system, it has been allocated four equal-sized memory blocks for the application program. For example, on the fourth block of memory to enlarge, which contains a part of the memory pool information and three equal size memory pool units. The unit 1 and unit 3 are free, unit 2 has been allocated. When application program need to allocate a unit size of memory through the memory pool, only need a simple traversal of all pool size information, then locate quickly the free memory pool block unit. Then according to the size of the block position information directly locate the first free unit address, return the address and mark the next free unit; Marking directly the corresponding memory unit of the memory pool size information is free when the application program release a memory pool unit.

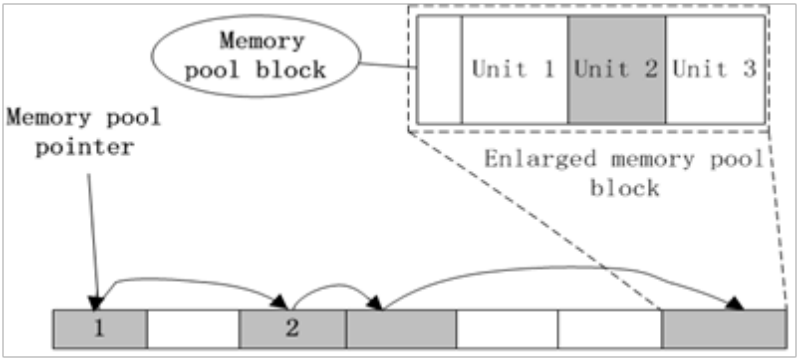

Figure.1 The Working Principle of Memory Pool

2.2 Small object allocation technology

Due to the application of memory block size of memory pool is uncertain, usually directly use the API of New and Malloc to apply for allocating memory. It is not effective for the small object allocation, when frequently used will cause a large amount of memory fragmentation and then reduce the performance, so the small object memory allocation technology[7] suitable for the small object memory allocation is used here.

The size and number of blocks can be set in the construction period of the small object distributor. The Chunk layer contains logical information, it can configured and returned the block from memory. Once there is no free block in the Chunk layer, the function returns zero. Small Object Pool layer contains a vector, Chunk objects stored inside, the Chunk layer has been extended. There is a chunk queue, which stores all the information, there are two Chunk pointers, one pointing to the currently available Chunk, one pointing to the current with the release of a pointer.

2.3 Scene analysis of search engine system

In this paper, analyzing the characteristics of the three scenes, Fixed length scene, Size is not fixed scene and Multiple allocation scene, designing the corresponding memory pool.

1) Fixed length scene

In the existing search engine system, cache design takes advantage of the hash tables, Original system use the New and Delete functions for the allocation and release of each node of the hash table, and the size of the node is fixed, according to the allocation and release of the fixed size nodes, a memory pool is designed to improve the speed of cache allocation and release. A lot of places use the Map of STL[9] in

the present search engine system, and the allocation and release of memory of each node in the Map is managed by the distributor in the STL, take over the fixed node memory allocation and release by itself, enhance efficiency, easy to debug.

Based on the above two scenarios, the common is that how to deal with node fixed size, design a small object dispenser to distribute and release the fixed size memory node.

2) Size is not fixed scene

In the cache management of the currents earch system, the search results are put into the cache, which is helpful for the next search, the size of each node in the cache is uncertain, and the time to enter the cache and propose cache can not be estimated. In the update module of the current search system, which manages the update and delete of the document, but the size of the document and the time of the update is unknown.

For this scene, it can design a recoverable variable length memory pool, the lock can be added to deal with base on the characteristics of cache multi thread[10].

3) Multiple allocation scene

The current search engine will return a result within 10M size after input a keyword, and a lot of information that comes with results will be allocated and released by using New and Delete function, it cause that the New function used frequently, and affect efficiency and bring memory fragments[11].

After analysis, the search engine return the result sat the same time, memory is frequently allocated, the number of release carried out only when the results of the query are returned, so the factor of frequent distribution should be considered, and the total capacity is not more than 10M, therefore, it is consider to allocate a large chunk of memory, after which all of the small memory is allocated, and finally released through the interface. Based on this scene design allocate not free memory pool.

3. Memory pool design and realize based on the high concurrency

Three scenarios are obtained by analyzing the current search engines: Hash table insert delete, Cache update and document update module, Query result return. Three memory pools are designed for the three scenarios: Recoverable Fixed Length Memory Pool and Recoverable Variable Length Memory Pool and Allocate Not Free Memory Pool were designed. The design structure of the search system memory pool is shown in Figure 2.

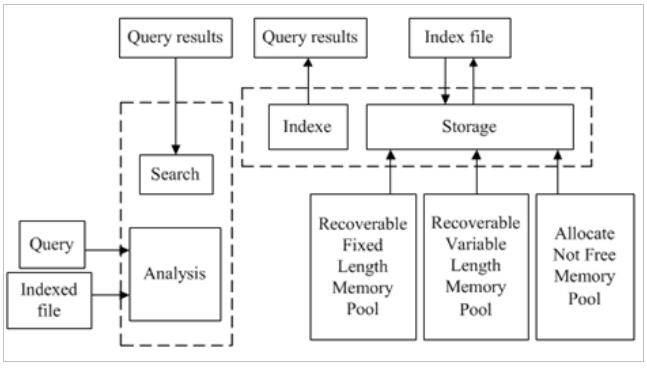

Figure.2 The Design Framework of Memory Pool of Search System

3.1 Recoverable Fixed Length Memory Pool

Recoverable Fixed Length Memory Pool is Small object distributor, it divided into 4 layers structure. As shown in Figure 3, the bottom layer is the Chunk object, each Chunk manages a large chunk of memory, which contains an integer number of fixed size block. Chunk contains logical information, the user can configure and return the block according to it. When the Chunk is no longer remaining blocks, the configuration fails and returns to zero. The second layer is Fix Allocate, which base on the first layer, using the known vector to expand the first layer and ensure that the size of the distribution can be extended. The third layer is Small Object Allocator, which provide universal distribution and return function. The third layer expand base on the second layer, it provide multiple second layer objects, it make the fixed length of the distribution technology turn into a variable length distribution technology. The fourth layer is Small Object, It made a package for the third layer, which provides a number of generic interfaces for the third layer and some common interface, extend it into a multi thread available distributor. Through layer by layer expansion, not only to ensure the release efficiency of the distribution, but also to better package the internal structure together, it not visible to the outside. By providing a common interface, to make it used like the operating system comes with the default memory.

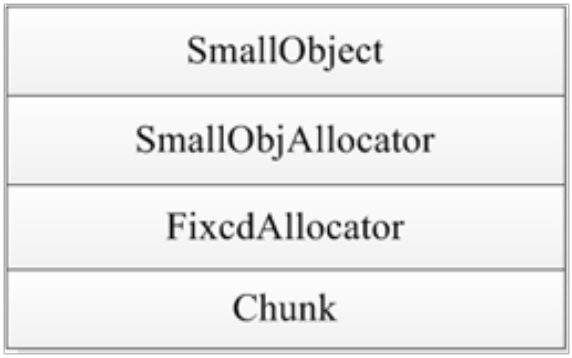

Figure.3 The Structure of Small Object Distributor

3.2 Recoverable Variable Length Memory Pool

Recovery variable length memory pool is a multi-threaded, variable length, recyclable memory pool, similar to hash table. A linked list indicates an assignable size range, each element in the list is a specific size of memory block pointer, which point to a list of memory blocks, to find specific head pointer by aligning, and then assign a node outside in the list. The elements in the range will be allocated through the New, when released, it will be returned to the pool for the next allocation, and beyond the range of elements also be allocated through New, but when released, it directly call Delete, and return to the operating system. About the factors of thread, add lock to ensure the thread safety after the specified by the constructor. Mainly includes Block Header layer, tragCtrlUnit layer and RecycleLitePool layer. The structure of the graph is shown in figure 4.

Block Header layer is the bottom of the distribution structure, nCtrlIndex indicates the size of block

distribution, pNextBlock indicates the next block, the structure is linked list structure, the whole structure is Union type, which save space and improve efficiency. The tragCtrlUnit layer is a headpointer of each BlockHeader layer, and also contains a member that indicates the number of BlockHeader objects. RecycleLitePool layer contains the thread element, the lock element, thetagCtrlUnit layer pointer, some count elements and a memory distributor, default for the New distribution and delete function to delete.

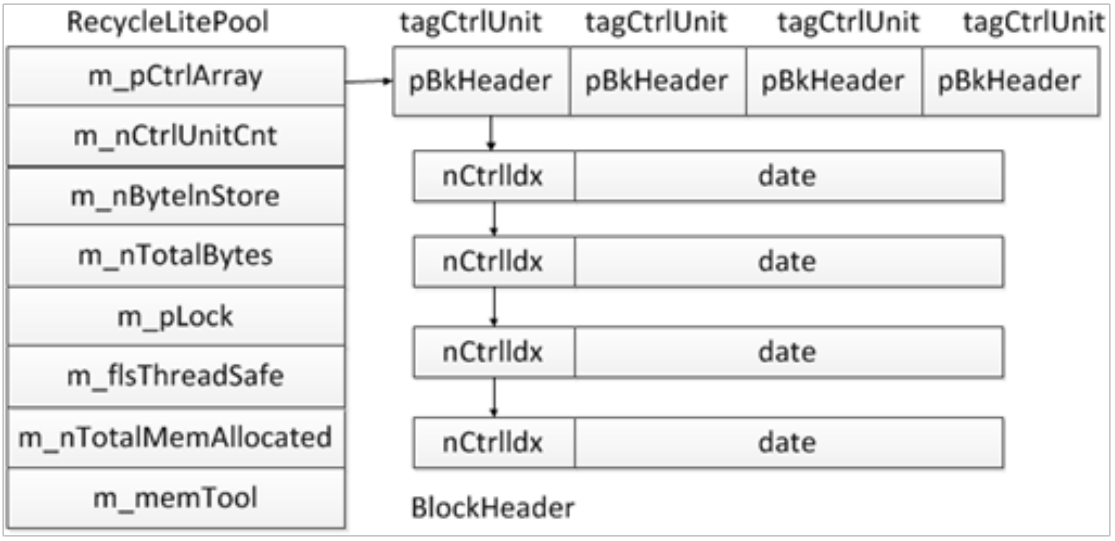

Figure.4 The Structure of Recyclelitepool

3.3 Allocate Not Free Memory Pool

Allocate Not Free Memory Pool is divided into four layers:

Memory Chunk layer: The bottom of the allocation block, there are three members inside, one indicates block size, one indicates location of initial address, one indicates currently available location.

Chain of Memory Chunk layer: Memory Chunk object is organized into a two-way linked list.

Simple Allocate Poilcy layer: This layer accept the request of distribution, change the size to be allocated not less than the size of Memory Chunk, and then added to the two-way linked list, the pointers of current distribution block point to the new block.

StagePool layer: This layer is the outermost layer, The default template parameter is Simple AllocatePolicy type, which provides external interface for distribution and release.

The overall structure of the StagePool contains a Chunk type pointer, which point to the currently allocated block, the allocation request are looked for from the current block every time, when the margin is not enough, it create a new block inserted into the list, select allocation strategy through the template parameter of Allocate Policy. The structure of the graph is shown in figure 5.

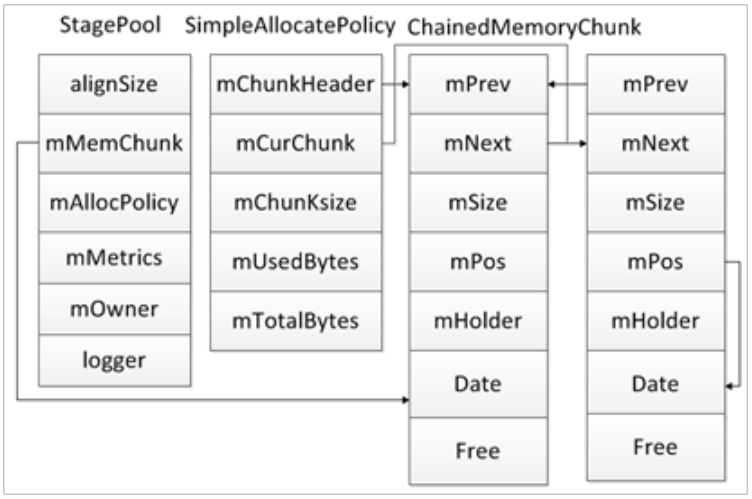

Figure.5 The Structure of Stagepool

4. Performance test and analysis

Through the Centos operating system, the compiler and debugging tools of vim, g++ and scons, Some scenarios are designed to simulate the actual scene of the search engine to test the performance differences between the default memory distributor and the designed distributor.

4.1 Performance test and analysis of the small objects distributor

1) For the small objects allocator test, when the amount of data is 100000, by testing the system function of New/Delete and the memory pool interface function of Allocate/Deallocate. Record 5 groups of data, as shown in Table 1, by analyzing and calculating the time difference, the small objects allocator is increased by 70.20% relative to New of system, and compared with the Delete, it is increased by 2.29%.

2) For the thread adapter hash table test, using the Node Allocator class to match the hash table, construct an identical class of Default Allocator, the internal partition function use New/Delete to achieve, using template parameters to match a hash table. For the single threaded test: For Node Allocator and Default Allocator, to applicate and release 50000 block, assuming that the program is running a fixed time of 10s, during this time repeated inserted and delete operation; Multi-threaded test: For Node Allocator and Default Allocator, to open the same number of threads, and execute thread function, to insert and delete data for corresponding hash tables. Table 2 indicates the test data of hash table. By analyzing and calculating: In the case of Single threaded, in terms of distribution, the efficiency of Node is increased by 22.61%, for the release, the efficiency of Node is reduced by 18.94%; In the case of multiple threads, in terms of distribution, the efficiency of the Node is increased by 13.80%, for the release, the efficiency of Node is reduced by 1.20%.

3) For the test of the small objects allocator adapter map container, to achieve a Small Object Alloator type, internal distribution released is achieved by Small ObjectPool, through the Map function, fit Small ObjectAllocator to Map, Map node distribution and release call interface of Allocate/Deallocate of Small ObjectPool; Similarly to achieve a NewAllocator type, internal distribution release is achieved by

New/Delete interface, also mapping to the Map through the constructed function; To compare with distributor type provided by the system. Single threaded test: Three map were inserted into 100000 data; Multi threaded test: Map data is inserted and emptied by three Map circulation within a certain time. Table 3 indicates the test data of adapter Map. By analyzing and calculating: In the case of Single threaded, in terms of Default, the efficiency is increased by 48.30%, for the New, the efficiency is increased by 54.50%; In the case of multiple threads, in terms of Default, the efficiency is increased by 35.90%, for the New, the efficiency is increased by 33.10%.

Table 1 Testing Data of the Small Objects Distributor(μ s)

Number of times	New allocation time	New release time	Small allocation time	Small release time
1	5318	2739	3174	2157
2	4662	1994	1271	1792
3	4856	2317	1350	1919
4	5093	3044	1381	1988
5	4899	1953	1253	1916
6	4768	2344	1329	3160
7	4835	2051	1743	1858
8	5852	2070	1388	1819
9	6412	2783	1328	1892
10	5119	2310	1271	1859
Average time	5181.4	2360.5	1548.8	2036

Table 2 Testing Data of Thread Adapter Hash Table(μ s)

Number of times	Single thread default allocation	Single thread default release	Single thread node allocation	Single thread node release	Multi-threaded default allocation	Multi-threaded default release	Multi-threaded node allocation	Multi-threaded node release
1	42	39	33	48	141	130	121	131
2	43	39	33	48	143	130	123	131
3	43	40	33	47	147	127	119	125
4	43	40	33	48	147	133	131	134
5	43	39	33	48	141	127	123	131
6	43	40	33	48	140	125	121	128
7	43	40	33	48	143	129	126	131
8	43	39	33	48	140	125	119	125
9	43	40	34	39	141	125	123	129
10	43	40	34	49	141	127	122	129
Average time	42.9	39.6	33.2	47.1	142.4	127.8	122.8	129.4

Table 3 Testing data of Adapter Map Multi Thread(μ s)

Number of times	Single thread Default	Single thread Small	Single thread New	Multi-threaded Default	Multi-threaded Small	Multi-threaded New
1	18	10	23	340	229	378
2	19	10	21	381	247	375
3	18	10	19	391	261	361
4	17	9	20	404	241	356
5	19	9	19	419	252	383
6	18	9	19	388	251	377
7	16	9	19	367	260	374
8	18	8	19	373	249	401
9	17	8	22	435	257	366
10	16	9	19	396	248	360
Average time	17.6	9.1	20	389.4	249.5	373.1

4.2 Performance test and analysis of recoverable variable length memory pool

Given a set of arrays with assigned sizefrom 1-10000, 4 threads, 5000 insert delete action, then use a variable length memory pool to assign and storage an array of pointers, to release and reallocate, cycle 20 times to get the test data results; Using Malloc to open the corresponding bytes of memory and assigning to another pointer array to storage. Under the same conditions, compare the time of distribution and release of the system function. Table 4 is the test data, by analyzing and calculating: In terms of New,the efficiency of RecycleLitePool is increased by 13.84%.

Table 4 Testing Data of Recoverable Variable Length Distributor(ms)

Number of times	New	RecycleLitePool
1	183	177
2	185	157
3	181	154
4	197	164
5	177	153
6	183	154
7	183	156
8	182	154
9	183	159
10	181	153
Average time	183.5	158.1

4.3 Performance test and analysis of allocate not free memory pool

Building a new distributor structure, alloc is interface of the distributor, using the space of distributor to allocate 4 bytes every time, allocated 10000 times; As a contrast, the system call the New function to allocate 4 bytes each time, to record the time of 50000 application action. Table 5 is the test data, by analyzing and calculating: In terms of New, the efficiency of StagepPool is increased by 90.80%.

Table 5 Testing Data of Allocate Not Free Distributor(MS)

Number of times	StagePool	Newl
1	4	51
2	5	47
3	4	48
4	4	48
5	5	48
6	5	48
7	4	48
8	4	47
9	5	47
10	4	47
Average time	4.4	47.9

5. Conclusion

1)Three scenarios are obtained by analyzing the current search engines: Hash table insert delete, Cache update and document update module, Query result return. Three memory pools are designed for the three scenarios: Recoverable Fixed Length Memory Pool and Recoverable Variable Length Memory Pool and Allocate Not Free Memory Pool were designed.

2)Using thesystem default memory management function, malloc/free and new/delete. By analyzing of the various factors of the function. Allocating and freeing memory on the heap increases overhead. The design of the memory pool is applied to the search engine system. It optimize the internal memory management and improve the search speed. For the test of the three memory pool, Compared with the system’s default memory, its efficiency are increased by 70.20%, 13.84%, 90.80%.

Sponsors or Supporters

This paper is partially supported by Special research project of Shaanxi Provincial Department of Education “16JK1376”.

Reference

[1] DAI Chunyan, XU Zhiwen. Discussion About Malloc/Free and New/Delete in C++[J].Science&Technology

of Baotou Steel (Group)Corporation, 2009(35):59

[2] LI Qian, PAN Minxue, LI Xuandong. Benchmark of Tools for Memory Leak [J]. Journal of Frontiers of Computer Science and Technology.2010(01)29.

[3] WANG Xiaoyin, CHEN Lijun. Implementation and Application of the Memory Pool in Linux Kernel [J]. Journal of Xi’an University of Posts and Telecommunications. 2011(04):40.

[4] QU Weihua, WANG Qun. Introduce and Analyzing of Search Engine Principle [J]. Computer Knowledgeand Technology. 2006(06):113.

[5] DUAN Bingying. Study and Design of Web Crawler in Search Engine [D]. Xidian University. 2014.

[6] GUO Bingxuan, ZHANG Jingli, ZHANG Zhichao. Algorithm of Spatial Data Scheduling Based on Memory Pool [J]. Computer Engineering. 2008,34(06):63.

[7] LIU Tao, NIE Xiaofeng, JING Jiwu, WANG Yuewu. Memory Management in Worm Simulation based on Small Object Memory Allocation Technique on The GTNetS [J]. Journal of Graduate University of Chinese Academy of Sciences. 2012,29(01):131.

[8] GUO Xufeng, YU Fang,LIU Zhongli. An Efficient Memory Built-in Self-Repair Method Based on Hash Table [J]. Acta Electronica Sinica. 2013(07):1371.

[9] LAI Xiangfang. Select The Appropriate STL Containers [J]. Digital Technology and Application. 2015(09):177.

[10] Alexandrescu A.Modern C ++ design: Generic Programming and Design Patterns Applied [M]. Boston: Addison-Wesley Professional. 2001.

[11] Robert W.P.Luka, Wai Lamb. Efficient In-Memory Extensible Inverted File [J]. Information Systems, 2007(32):733.

AuthorBrief

Liu Ping-Ping(1971-), female, Associate Professor, Xi’an Technological University, Research area: Artificial intelligence